The Journal of Systems and Software 61 (2002) 129–143

The Journal of Systems and Software

www.elsevier.com/locate/jss

Visual requirement representation

Deng-Jyi Chen [a,*], Wu-Chi Chen [a], Krishna M. Kavi [b,1]

[a] Department of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC

[b] Electrical and Computer Engineering Department, The University of Alabama at Huntsville, Huntsville, AL 35899, USA

Received 31 March 1999; received in revised form 30 April 1999; accepted 5 April 2001

Abstract

Multimedia technology has played an important role in modern computing because it offers more natural and user-friendly interactions with an automated system. This is particularly true for systems utilizing graphical, icon or window-based input and output. Multimedia technology also facilitates "reuse" more naturally, since the basic components and functions of presentation and animation can be reused for several different animation scenarios. This is evidenced by the rapid prototyping capability of computer and video games where although the characters and story lines change, the basic animation remains constant. In this paper we utilize multimedia technology for eliciting requirements of software systems, particularly those systems that utilize windows- (or graphical)-based interactions with the user. Our methodology will implicitly emphasize reuse since in our approach reusable components include not only code and documents, but also voice narration, animation sequences and message mechanisms. We call such software components as multimedia reusable components (MRCs). Using MRCs, one can view software requirements instead of reading textual representation of the requirements.

Keywords: Multimedia; Requirement scenario; Visual requirement

1. Introduction

An important phase in the software life cycle focuses on eliciting the requirements from users (Pressman, 1992). Gathering requirements is an extremely difficult task. In performing this task, five requirement elicitation problems must be addressed: completeness, communication, irrelevancy, incorrectness, and inconsistency. Users and engineers generally have no common terminology or domain knowledge while establishing the requirements for the software development. As widely recognized, ineffective communication frequently leads to misunderstandings and irrelevant or incorrect requirements. Also, different users offering varied perspectives lead to inconsistent requirements. Furthermore, voluminous textual requirement documentation arising from requirements gathering process is typically difficult to comprehend.

Multimedia technology has played an important role in modern computing because it offers more natural and user-friendly interactions with an automated system. This is particularly true for systems utilizing graphical, icon or window-based input and output. Multimedia technology also facilitates "reuse" (Lenz et al., 1987; Freeman, 1987; Lubbars, 1987; Maien et al., 1995) more naturally, since the basic components and functions of a presentation and animation can be reused for several different animation scenarios. This is evidenced by the rapid prototyping capability of computer and video games where although the characters and story lines change, the basic animation remains constant.

In our ongoing research we utilize multimedia technology for eliciting requirements of software systems, particularly those systems that utilize windows- (or graphical)-based interactions with the user. Our methodology will implicitly emphasize reuse since in our approach reusable components include not only code and documents, but also voice narration, animation sequences and message mechanisms. We call such software

* Corresponding author. Tel.: +886-35-712121 x 3701; fax: +886-35-724176.

E-mail addresses: djchen@csie.nctu.edu.tw (D.-J. Chen), wjchen@csie.nctu.edu.tw (W.-C. Chen), kavi@cs.unt.edu (K.M. Kavi).

[1] Present address: Department of Computer Science, The University of North Texas, P.O. Box 311366, Avenue B at Mulberry, Room GAB 320, Denton, Texas 76203.

0164-1212/02/$ - see front matter
PII: S0164-1212(01)00108-X

components as multimedia reusable components (MRCs) (Chen, 1998). Using MRCs, one can view software requirements instead of reading textual representation of the requirements. It is not our intention to state that one should completely eliminate textual representations, particularly if the textual representation leads to formal analysis. We also concede that visual requirements are not readily amenable to formal analyses since they present a subset of possible scenarios. However, we feel that multimedia-based approach to requirements elicitation has its place in Software Engineering practice, since this presents users a visual effect and permits early feedback on the requirements. In many cases this form is more natural for communication between the Software Engineer and the customer. Such a visual (or artistic) form of communications is very common in many human and societal interactions (e.g. architecture and fashion industry).

In this paper, we concentrate mainly on requirement acquisition, particularly the communication ability, requirement representation, presentation, and organization based on multimedia reusable components. A visual requirement representation model is also proposed, along with an authoring tool based on that model. Requirement reuse has been studied for years (Maien et al., 1995; Lam et al., 1997; Massonet, 1997; Keepence et al., 1995). Multimedia reusable components for requirement reuse provide another natural way to achieve the reuse of requirements. Although MRCs facilitate reuse, we do not emphasize this aspect of MRCs in this paper.

2. Visual modeling

2.1. Visual modeling

James Rumbaugh stated "modeling captures essential parts of the system" (Rumbaugh et al., 1987). Modeling is extensively employed during software requirement analysis and design (Freeman, 1987; Booch, 1991; Coad and Yourdon, 1990). In visual modeling we use standard multimedia notations to describe a system's requirements and software programs. Visual modeling can capture a system's functionality from the user's perspective. We feel that such an approach is a natural communication form and can be used to capture the system objects and logic from users. Visual modeling can represent levels of system abstraction, thereby permitting the management of the system complexity. In a visual modeling environment, multimedia notations are repeatedly used, thus encouraging software reuse. Conventionally, requirement acquisition process originates from introspection, questionnaires, interviews; those results are then represented in textual form. Problems arising from this process have been widely addressed (Goguen, 1996). Of recent interest, an increasing number of video and multimedia approaches have been proposed for requirement elicitation (Jirotka et al., 1995; Wood et al., 1994; Ohnishi, 1994; Takahashi et al., 1996; Breen et al., 1987). For instance, a Video-based Requirements Elicitation project was studied at Oxford University (Jirotka et al., 1995). A project entitled "advanced multimedia organizer for requirements elicitation (AMORE)" at Carnegie Mellon University is developing a modeling tool for assisting with requirements elicitation using multimedia technologies (Wood et al., 1994). A Visual Software Requirements Definition approach was proposed at Kyoto University (Ohnishi, 1994). In this approach, standard shapes and semantics of icons are defined for constructing a requirement scenario prototype. Also provided in this approach is a mechanism for detecting discrepancies in the elicited requirements and user's wishes. Finally, NTT Software Laboratories use Hypermedia support for Collaboration in Requirements analysis (Takahashi et al., 1996).

In contrast with conventional requirement engineering, applying multimedia approach as an innovative way for requirement representation and presentation to software engineers is emphasized in this paper. The major difference between the visual programming model proposed here and other visual language tools (Hirakawa, 1987; Chang, 1995; Burnett et al., 1995; Baroth, 1995) is the ease of programming. We use the multimedia reusable components as the basic elements in our visual programming model. Multimedia reusable components itself is like an actor in the showroom and our visual language is used to create a scenario for those actors to create a performance (animated scenario requirement). Thus, our visual programming model deals with programming in "super-large" (in the sense that MRC is more than a class-based programming) in the visual way. Also, the proposed tool provides designers capability to create an animated visual requirement. This is what we do not see in other related research.

2.2. Equivalence of requirement representation model

High level requirements can be represented using textual or visual description and these high level models are then transformed into requirement specifications (or called determined requirements), as depicted in Fig. 1. Assume that we define $\boldsymbol{R}$ as the user requirement, $\boldsymbol{S}$ the requirement specification, $\boldsymbol{T}$ the textual model and $\boldsymbol{M}$

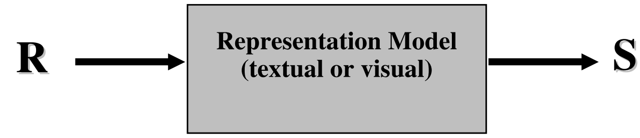


Fig. 1. Representation model.

the visual model. Ideally the visual representation of the requirement should be a *conceptual equivalent* to the textual representation

$\boldsymbol{R}(\boldsymbol{T})^{\circ} > \boldsymbol{R}(\boldsymbol{M})$ for the same requirement $\boldsymbol{R}$.

Restated, regardless of representation used, the user requirement must be the same. Note that we define the *conceptual equivalent* in terms of high level abstraction; in some cases visual models may be difficult to use for detailed description of user requirements. On the other hand, our preliminary experiments show that visual representation of high level requirements are preferred to textual representations, in terms of eliciting user requirements (Chen, 1998).

3. Visual requirements representation model

3.1. Using multimedia reusable components

MRCs are encapsulated as object-oriented paradigm and designed in a standardized format (Li, 1992; Chen, 1998). An MRC, which consists of multimedia data, operations, and message mechanism, can be viewed as a live (active) object of the system. Combining several MRCs allows us to produce a multimedia film. Assume that each MRC represents a requirement in a multimedia form. Then, each scene subsequently produced represents a requirement scenario and the complete film represents the requirements for the software systems being constructed.

3.2. Scenario

A *Scenario*, occasionally referred to as a *use case*, is a useful way of understanding the interface between the environment and the system. A software system and its environment can be extremely complex, and capturing the behaviors and relationships among subsystems is often difficult with textual descriptions. This leads to misunderstandings between users and developers. On the other hand, scenario is an evolving description of situations in the environment. The use of scenarios can more readily elicit correct (or desired) behavior of the software, and can clarify the interrelation between functional and non-functional requirements. A repeated scenario can be reused as a requirement pattern. Here we propose visual scenarios using multimedia and visual effects to represent complex behaviors and interactions. In this manner, a software requirement is executed with a scenario of multimedia presentations.

3.3. The proposed model

The visual requirement representation model, as shown in Fig. 2, includes six parts.

Visual requirements authoring system (*VRAS*) can be considered as a multimedia authoring tool, consisting of a script language, graphical user interface (GUI) for interconnecting MRCs, and animating them. It is used to capture a requirement visually and playing it back to the customer.

Customer: A user who provides the high level requirements for the system and may participate in creating the requirements film (or movie) using the VRAS.

System analyst: The system analyst can use VRAS to capture a customer's requirement. An analyst must also determine which MRC is necessary for the customer's requirement.

Component constructor: When existing MRCs are not adequate to describe user requirements, a new MRC will be added to the MRC manager using the component constructor. The constructor includes a multimedia

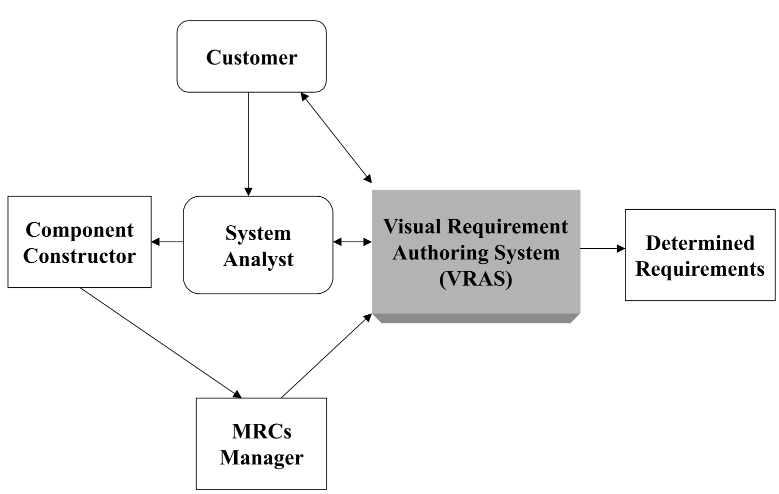


Fig. 2. Visual requirements representation model.

editor and a relatively simple programming environment to encapsulate multimedia data and related code components. Multimedia artists and the software engineer are involved in MRC construction phase.

MRCs' manager: A database management system for managing MRCs and provides interface for adding or deleting MRCs as well as for retrieving an existing MRC. The manager organizes and stores MRCs (along with all related files) based on component standardization information.

Once the visual requirement is produced, it is possible to transform the requirement into a formal form. We call these determined requirements, and the determined requirements dictate the design and coding phases. In this paper we will not describe automated transformation of visual requirements into determined requirements.

3.4. Definition of terms

Four basic authoring elements are available to create a visual representation of a requirement: scenes actors, relationships, and window-based operations.

Actors: An actor, which is an object that can send or receive messages, represents a requirement segment and is a minimal reusable component for requirements. In our case each MRC will be considered an actor.

Scenes: A scene is a container where an actor is used to represent a requirement session. A scene can therefore be used to accumulate one or more requirement scenarios and a requirement scenario consists of one or more MRCs. A scene also defines a requirement framework. In a scenario, an MRC may be replaced by another MRC. A scene itself can become a reusable component of a requirement framework.

Table 1
Presentation actions and script language features

Presentation requirement	Actions	Requirements presentation meaning	Script language features
Scene background and display effect	Select display effect. Select background. Save background. Draw on background	Beautify the look of a scene	Background (ID, file name). Effect (ID)
Actor arrangement	Arrange actors' position and animation movement on visual editor. Set actors' attributes	Basic requirement components	Actor name (ID, file name). Position (x, y). Size (width, height). Speed (frames). Depth (level). Rotate (degree). Path $((x1, y1), (x2, y2), \ldots)$
Sequential presentation	When an anchor actor starts, a list of actors will follow its presentation one by one. A finished message is returned when the last actor in the list finishes its presentation	Sequential scenario. Represent serial message flow, control flow, or processes	Sequential (anchor, actor list)
ParallelOR presentation	When an anchor actor starts, a list of actors also starts their presentation at the same time. A finished message is returned when any one of actors in the list finishes its presentation	Parallel scenario. Concurrent processes. Control transfer occurs at any of the current processes terminated	ParallelOR (anchor, actor list)
ParallelAND presentation	When an anchor actor starts, a list of actors also starts at the same time. A finished message is returned when all actors in the list finish their presentation	Parallel scenario. Concurrent processes. Control transfer occurs at all of the current processes terminated	ParallelAND (anchor, actor list)
Delay presentation	Delay period after receiving the start presentation message. It is used with sequential and parallel presentation	Represent a real-time constraint	Delay (actorID, second)
Loop presentation	Repeatedly present the actor	A process is continuous running or repeats several times	Loop (actorID, times)
Appear/hide	Show an actor on a scene. Hide an actor from a scene	Represent creation or deletion of an object while process running	Appear (actorID). Hide (actorID)
Scene branch	Close current scene and open another scene	Scenario connection. Link to subsystem. To view a scenario in detail	LinkScene (sceneID)
Concrete	Define a scenario pattern	To represent a scenario (framework) for which actors in the scenario can be substituted	Concrete: Scene Begin ... Scene End
Start/stop	Start/stop presenting a scene	Start/stop visual requirements scenarios presentation	Start: Start (sceneID) Stop

Relationship: A relationship describes the intra-actor relationship and scene-actor relationship. The object relationships denote the presentation sequence among actors, e.g. serial or parallel presentation. The space relationships describe MRC's size, position, movement path, depth, and rotation when the MRC is placed in a scene. The time relationships describe the timing constraints during presentation including the presentation speed, loops, or delays. MRCs are organized by relationship features to execute a requirement scenario.

Requirement project: A requirement project organizes scenes to present a system requirement. The project can be viewed as a multimedia program and capture the interaction of actors in each scene.

These four elements can be utilized to author visual requirements or MRCs.

3.5. Script language

A scripting language is necessary to describe the connections among the MRCs and to define presentation behavior. In our system the scripting language (Chorng-Shiuh, 1995; Chen, 1998) is based on visual forms so that a user can create visual requirements easily, without having to learn a complex scripting language that is based on textual commands. The visual scripts are automatically converted into textual scripts.

While actors can only execute simple tasks, a script controls the MRC at a very high level. A script plays a role similar to that of a process manager in multitasking operating systems. Scripts create, delete actors and trigger events. The script and the actors execute concurrently.

A presentation script has two components: declarations and actions. The declaration part deals with the storage of objects. The action part concerns itself with the way these objects are presented. An action can be either primitive or compound. A primitive action is defined as a single media presentation. The language does not specify how primitive actions are executed since such actions are contained within MRCs.

The script language is used to describe the manner in which actions are executed, such as in parallel or in serial and their duration. Our script language handles the following relationships:

1. *Actor space relationship*: presentation path, icon size, position, depth, and rotation.
2. *Actor time relationship*: sequential, parallel-or, parallel-and, delays.
3. *Presentation properties*: speed, loop, appear, hide, start, stop.
4. *Actor creation and deletion.*
5. *Multimedia hyperlink between scenes*: jump, conditional branch.
6. *Concrete*: define a scenario pattern.

Table 1 summarizes the presentation actions and script language features.

4. Visual requirements authoring

4.1. Representation structure

Requirement scenarios are complex for a large software system. A structural organization for such a complex requirement scenarios should be provided. In our visual representation model, at least one scene represents the visual requirements. A scene includes actors and relationships to represent a requirement scenario. Organizing the requirement scenarios involves structuring the scenes. Scenes can generally be structured as a list, tree, or graph (see Fig. 3) (Chorng-Shiuh, 1995; Chen, 1998). Hyperlinks are used to connect scenes. These three basic scene structures can be arbitrarily combined to describe a complex requirement scenario. A specific scene organization can be abstracted into a reusable requirement representation pattern or representation framework.

4.2. Visual requirements authoring

A visual requirement is created by visual operations and captured by the script language. Fig. 4 illustrates the visual requirements authoring process. We initially create a project for the system. Next, one or more scenes can be created for the project. We can reuse a scene if a reusable scene pattern (application framework) is located. Otherwise an empty requirement framework is created. For every scene, actors (MRCs) are selected from MRC database and placed on the scene. For all selected actors in the scene, we define their actions and describe relationships to perform a scenario. A visual requirement scenario is then produced. The scenario can be previewed (or played) to examine whether the scenario meets the user requirement or not.

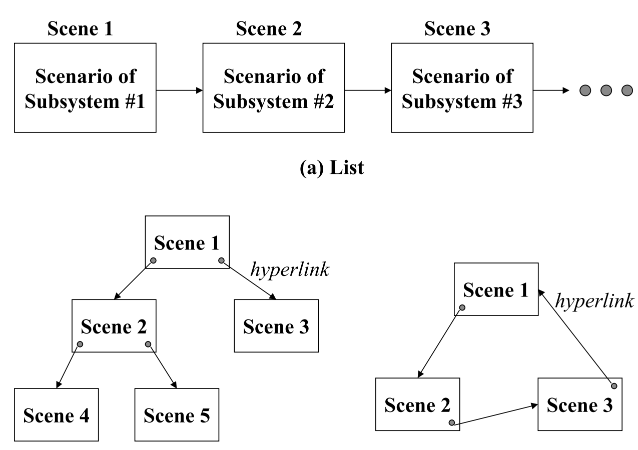


Fig. 3. Representation structure.

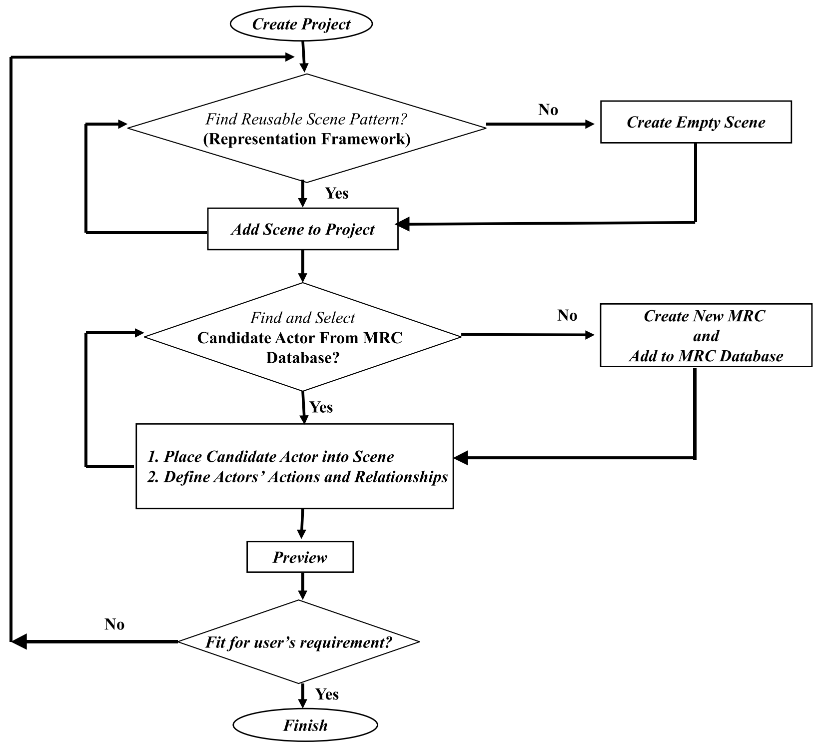


Fig. 4. Visual requirements authoring process.

4.3. Visual requirements reuse

The basic reusable component in our visual requirements methodology is an MRC. A scenario is a requirement segment. A specific scenario pattern may also be abstracted into a reusable requirement scenario pattern, since it provides a means of replacing component MRCs to produce a new requirement scenario. A scene represents one or more scenarios and the scene itself can be viewed as a framework for reusable object. A project describes an application framework – entities in the framework can be replaced by other entities to represent a different application framework. We contend that such a reuse during requirements elicitation can significantly enhance the software development.

5. Implementation

5.1. System structure

The major component of our visual requirements approach is the visual requirement authoring tool (VRAT), which allows users to select MRCs, stored in the MRCs Manager System, and creating a film (visual representation) that can be played to customers.

The MRCs must be designed in a standard format to permit searching and archiving MRCs for reuse. The MRC designers must seek assistance from artists in drawing meaningful and simple animations to accurately depict the basic meaning of an event (a requirement scenario). The authoring tool must provide functions that allow analysts to change an MRC's attributes, to create an animation sequence for an event in an MRC, and assemble MRCs together as a scenario-based requirement. These scenario-based requirements are then combined as a feature presentation (film) to be played by a playback system to users for the purpose of evaluating if the requirement representation satisfies his or her need. Fig. 5 depicts the system architecture.

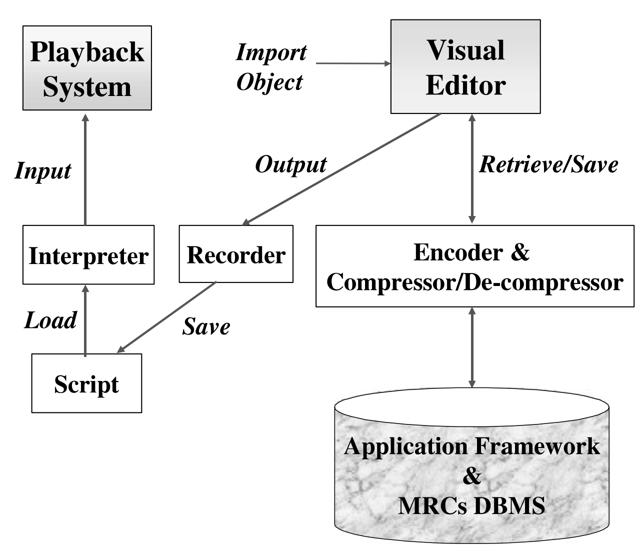


Fig. 5. System structure.

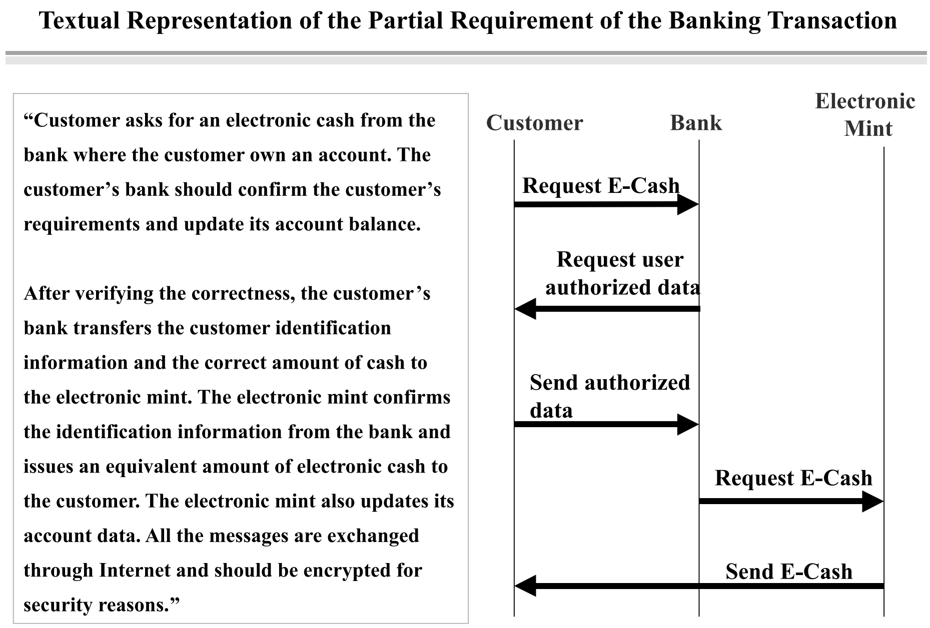


Fig. 6. Event diagram of obtaining scenario.

5.2. Example – A part of the banking transaction system scenario

In this section we show how our VRAT can be used to create a visual requirement. This example will be used to evaluate the differences between textual representation and visual representation.

After reading the description, the textual representation of the scenario is shown below:

1. Customer sends a "Request_For_E-Cash"message to the bank.
2. Bank requests the customer's personal data for authorization.
3. Customer sends his account information to the bank.
4. Bank updates the customer's account information.
5. Bank sends "Request_For_E-Cash"message to Electronic Mint.
6. Electronic Mint updates the bank's account information.
7. Electronic Mint sends "E-Cash" to the customer.

Fig. 6 depicts the event diagram of the textual scenario.

5.2.1. Visual representation of the partial requirement of banking transaction [2]

For the same requirement scenario shown in the previous segment is depicted using the visual requirement representation. Fig. 7 depicts the system behavior and the event diagram for the same requirement scenario in animated form. Notably, the circled number in Fig. 7 represents the presentation order. All objects are presented with animation with multimedia effects.

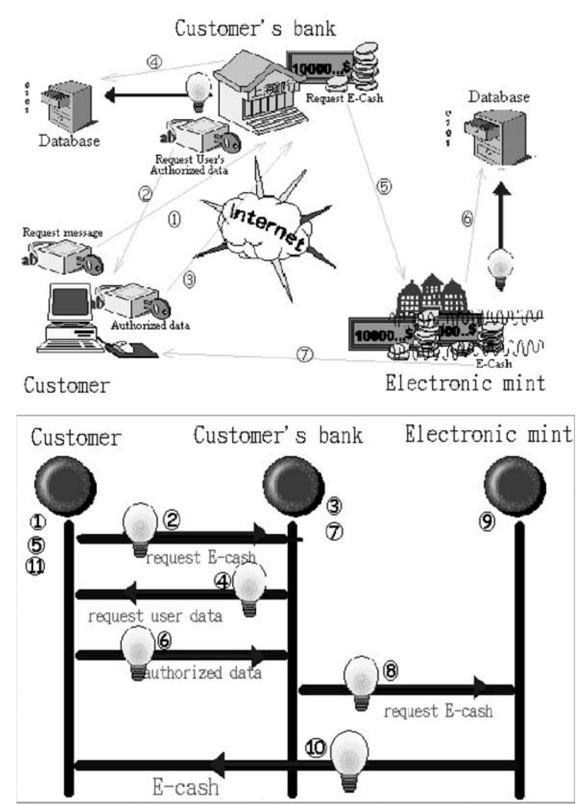


Fig. 7. Visualized event diagram of the requirement scenario.

[2] A copy of the CD that contains the animated requirement representation of Fig. 7 is available by request.

5.3. Evaluation of the proposed visual requirement tool

This paper outlined a visual requirement representation paradigm and a VRAT for creating visual requirement representation. To evaluate the effectiveness of the proposed VRAT, an experiment was performed. This experiment attempts the following:

1. To understand the difficulties involved in creating the visual requirements representation.
2. To evaluate the merits and limitations of the VRAT.

5.3.1. Experimental design

Twenty-four graduate students from a graduate course on Object-oriented Computing served as the control group. They were asked to use both the textual representation and the visual representation to represent a same problem. We collected the time needed for both approaches and collected responses to a questionnaire after the students completed the requirement representation.

5.3.2. Variables

The experiment measured four independent variables:

1. time usage;
2. degree of problem expression ability;
3. communication ease between users and developers;
4. understandability.

Variables 2–4 are measured in five ranks: 0%, 25%, 50%, 75% and 100%.

5.3.3. Design

This experiment compared the use of the textual requirement representation method and the visual requirement representation method. In light of the example, viz., electronic cash system (ECS) requirements, all subjects were asked to represent the requirements in both textual form and visual form. Appendix A describes electronic cash system. The requirement analysis should be represented by object modeling, dynamic modeling, and functional modeling. The object modeling includes three parts: identify object classes, data dictionary, and object model. The dynamic modeling consists of five parts: behavior description, scenarios, event flow diagram, state diagram, and data flow diagram.

Appendix A was given on the first day of the experiment, and not prior to the start of the experiment. The pictorial representations of the bank, database, electronic mint, etc. created as MRCs were provided as *.gif or *.bmp files. Thus all students in the experiment had equal footing for creating the "textual requirements" and "visual requirements". For the VRAT users group, as planned most of the MRCs related to the problem description could be downloaded to the VRAT's MRC database. Note that our claim is that the availability of such MRCs will make it easier to create visual requirements (without having to learn complex visual languages or authoring tools).

The time needed for using textual and visual representation was recorded. A questionnaire was designed for accumulating the subjective reflections of subjects after they had completed the textual and visual requirement representations. Appendices B and C list the results of the questionnaires. In the experiment subjects were divided into two groups randomly, group A and group B, with 12 persons per group. Processes of the experiment included four phases.

Phase 1: All subjects were requested to represent the system by using the textual model and the time spent was recorded.

Phase 2: Group A was asked to use the proposed VRAT to create requirement representation. Meanwhile, group B selected other multimedia authoring tool familiar to them, to author the system. Time needed was recorded by group B also.

Phase 3: Complete the questionnaires.

Phase 4: Accumulate and analyze the data.

Experimental results are reported in the following sections.

5.3.4. Data and analysis

5.3.4.1. Multimedia authoring tools used. Fig. 8 displays the multimedia authoring tools used in the experiment. One-half of the experimental students were asked to use

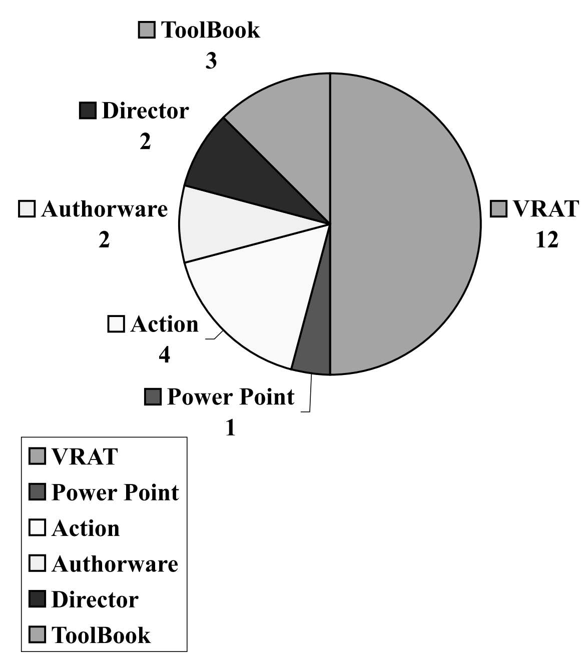


Fig. 8. Multimedia authoring tools used in the experiment.

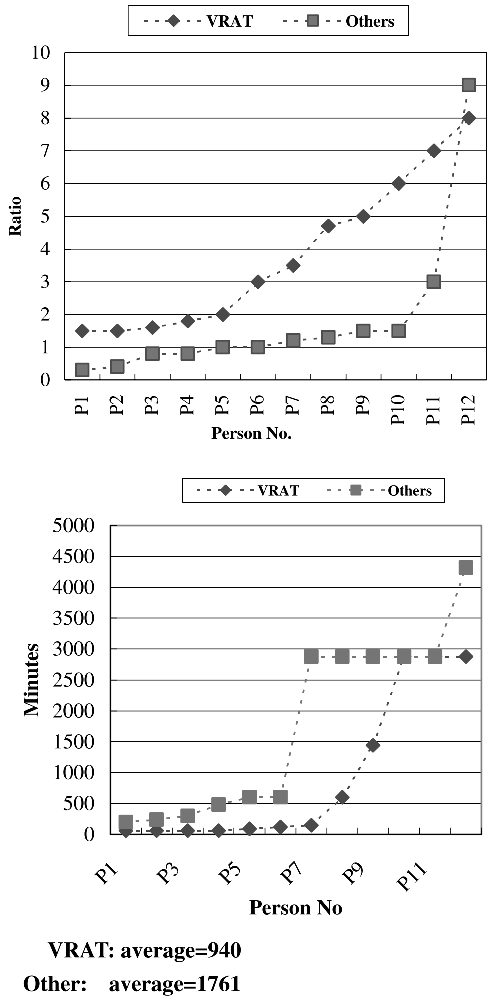


Fig. 9. Time ratio and time usage.

proposed VRAT. The other students selected their familiar multimedia authoring tools: Power Point, Action, Authorware, Director, and ToolBook. These five tools are referred to as "others" tool in this paper.

5.3.4.2. Time usage. The time ratio is defined as

$$\text{Time ratio} \equiv \frac{\text{time} - \text{usage in textual representation}}{\text{time} - \text{usage in visual representation}}.$$

Fig. 9 indicates that using the proposed VRAT, students spent less time for creating the same requirements, as compared to using other multimedia authoring tools. However, the time needed to create a meaningful MRC for matching a requirement under consideration was not including in the time estimates. Thus, this experiment applies to the case when sufficient number of MRCs is available in the database, the authoring of a visual requirement is relatively easy.

5.3.4.3. Representation difficulties. Figs. 10 and 11 compare the difficulties in using the proposed VRAT

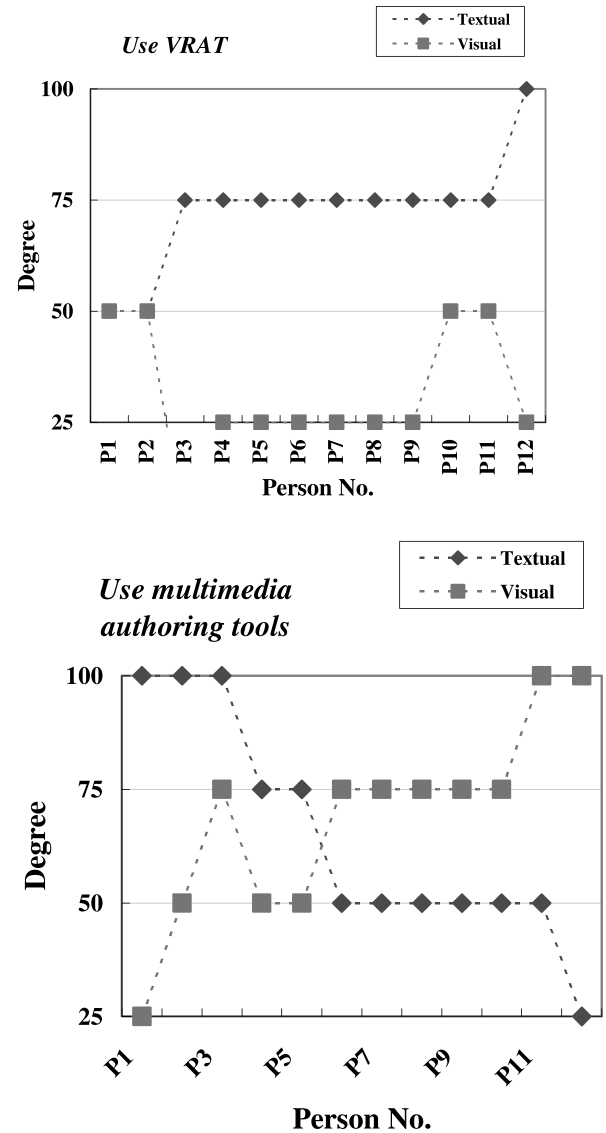


Fig. 10. Representation difficulties.

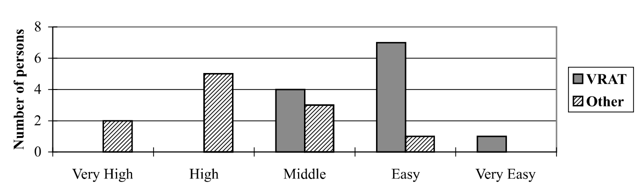
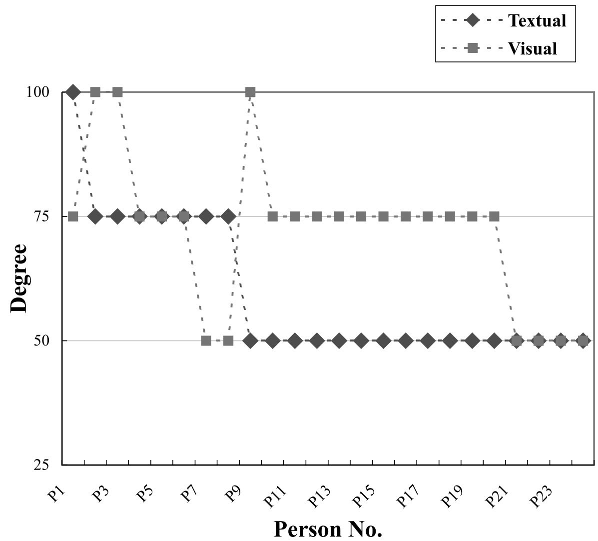
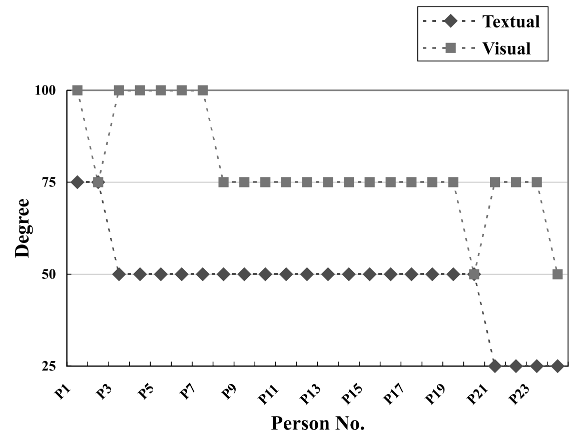


Fig. 11. Representation difficulties.

and other multimedia authoring tools. These figures show that VRAT is relatively easy to use compared to other multimedia authoring tools.

5.3.4.4. Problem expression ability. Fig. 12 shows that visual requirements representation expresses the problem better than the textual representation.





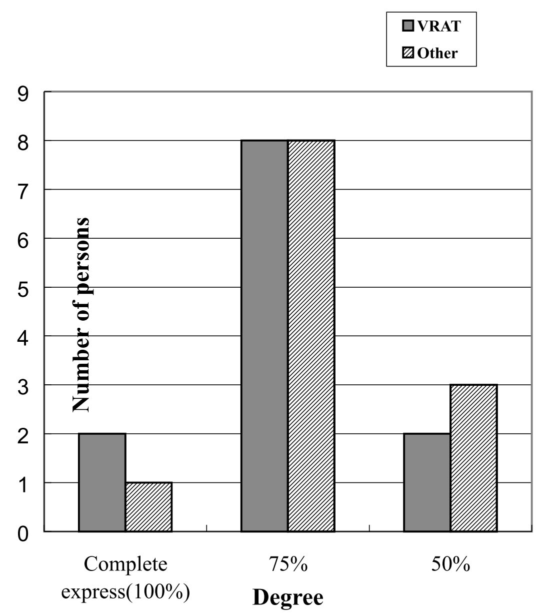


Fig. 12. Problem expression capability.

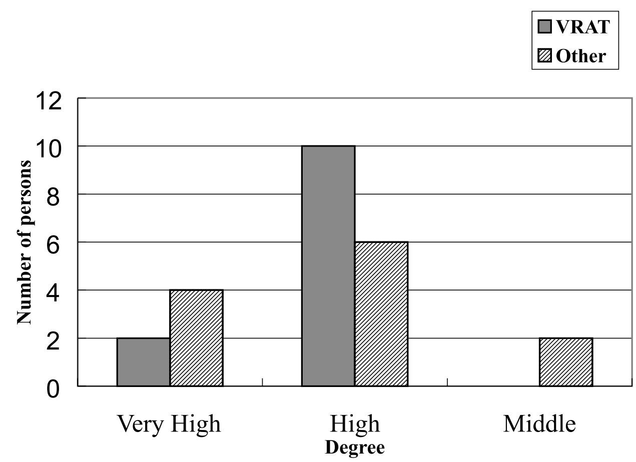


Fig. 13. The easiness of communication between users and developers.

5.3.4.5. The easiness of communication between users and developers. Fig. 13 compares the ease of communication between users and developers using our VRAT and other multimedia tools. The results indicate that VRAT facilitates more acceptable means for communication among users and developers.

5.3.4.6. Early feedback. Fig. 14 shows the degree of early feedback facilitated by our VRAT, other visual authoring tools and textual representation of requirements. The figure clearly shows the advantage of visual representations, and among visual representation, the advantages of VRAT.

Based on the results of the experiment, the following conclusions can be made:

1. Visual requirement representation provides an earlier feedback from the users.
2. Visual requirement representation facilitates for better communication between users and developers.
3. Visual requirement representation is more expressive in describing user wishes (user's requirement) because of the audio and visual effects.
4. Our requirement authoring tool with the multimedia features aids in the development of visual requirements and the presentation of the requirements to users.

As can be seen from the results of the experiment, the availability of multimedia icons (or MRCs) based on the domain can aid in the creation of visual requirements, animation of the requirements and early feedback from the customer to avoid any misunderstanding of the user requirements. The easiness of communication between users and developers variable is measured based on the questionnaire E in Appendix C. In our experiment students take the role of both users and developers; students in each group assume these roles and conduct necessary interviews. This experiment is limited in scope and a more extensive experiment is needed to further validate these conclusions.

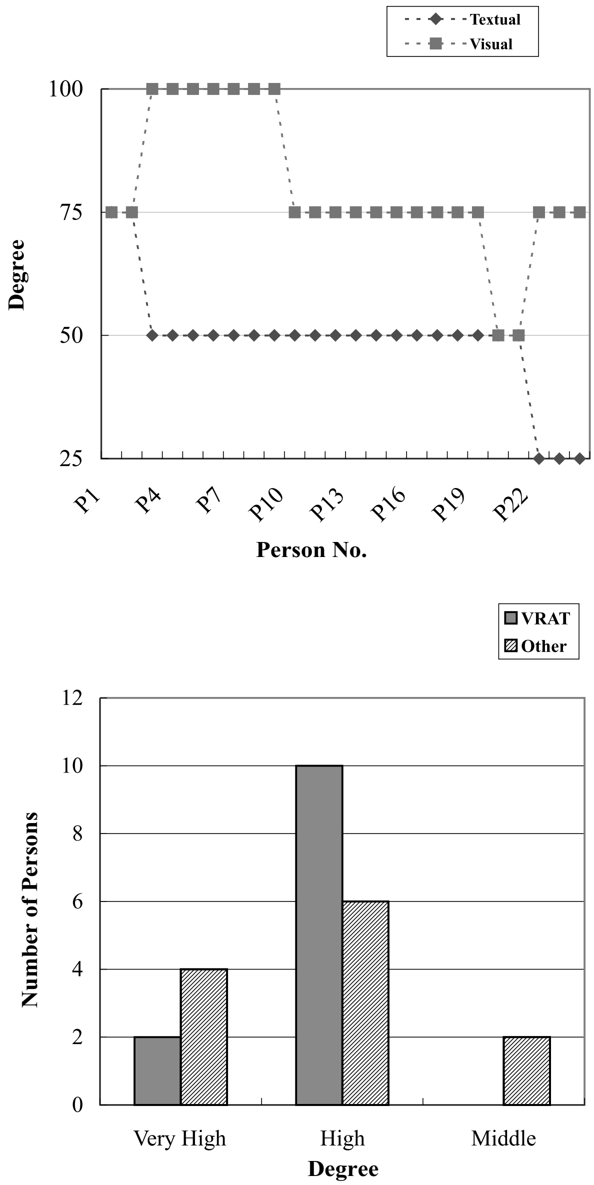


Fig. 14. Early feedback.

6. Conclusion

Our research shows that requirements can be visualized. The proposed MRCs make the visual requirement representation possible. By using visual requirements, the requirements can be viewed as an animation sequence instead of reading voluminous requirement documents. Such a novel software requirement representation technique facilitates for more natural means of communication between the users and the developer, provides early feedback from users, more expressive in describing user's wishes. Our requirement authoring tool with the multimedia features helps users to develop visual requirement and presentation.

Appendix A. Electronic cash system, E-Cash system [3]

A.1. System requirement

The E-Cash system provides trading methods and instruments for computer network users and merchants. Users can trade with merchants for goods and services through Internet, and complete the transfer of money without the flow of cash.

The following are involved in the operation of the system:

Customer
Merchant
Customer's Bank
Merchant's Bank
Electronic Mint

The operating procedure of the system is illustrated below. The direction of the arrow indicates the flow of information; the text besides the arrows display the information contents and the order of the message flow. All the messages must be protected for confidentiality. All the E-Cash transactions will be effective only after being approved and recognized by the Electronic Mint.

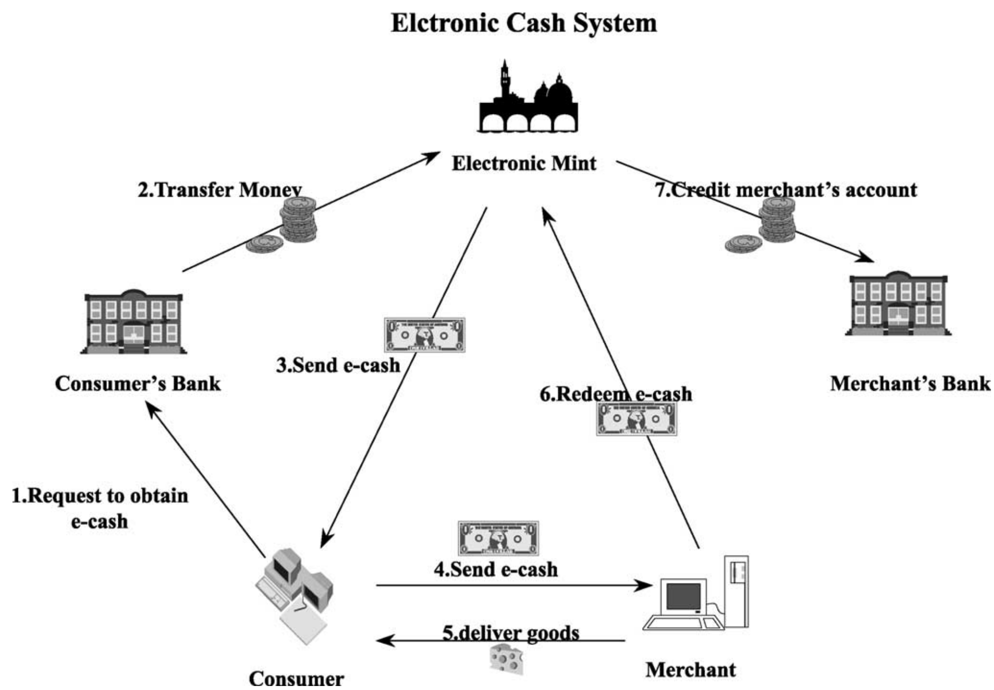
The system must complete three phases in each trade as described here.

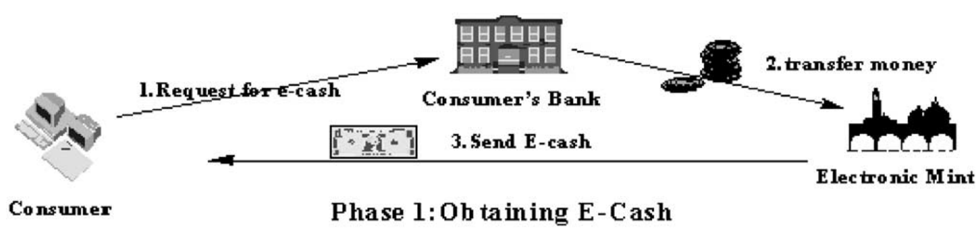
Phase 1: *Obtaining E-Cash*. Customer requests the E-Cash transfer from his/her bank. Customer's Bank insures sufficiency of funds, and transmits relevant customer's information (address, name...) together with amount of funds to be delivered to the Electronic Mint. The Electronic Mint sends E-Cash to the customer after receiving and crediting the funds from the customer's bank. The customer, his/her bank and the Electronic Mint must update their database to reflect the transaction.

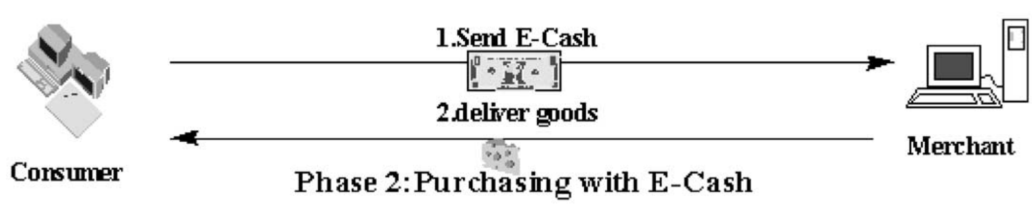
Phase 2: *Purchasing with E-Cash*. Customer can purchase goods from a merchant using the E-Cash delivered by the Electronic Mint. The merchant delivers the purchased goods to the customer after receiving the E-Cash.

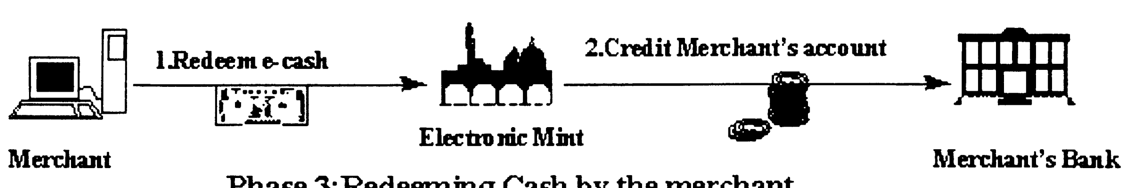
Phase 3: *Redeeming cash by the merchant*. The merchant redeems the money from Electronic Mint by tendering the E-Cash. The Electronic Mint deposits equivalent amount of money in the Merchant's Bank. The account information and the amount of funds to be credited will be exchanged between the Electronic Mint and the merchant's bank, and the two entities update their databases to complete the transaction.

[3] Please note that Appendices A, B and C were written in Chinese and were freely translated to English.


Elctronic Cash System
Electronic Mint
2.Transfer Money
7.Credit merchant's account
Consumer's Bank
3.Send e-cash
6.Redeem e-cash
Merchant's Bank
1.Request to obtain
e-cash
4.Send e-cash
5.deliver goods
Consumer
Merchant


1.Request for e-cash
Consumer's Bank
2.transfer money
3.Send E-cash
Electronic Mint
Consumer
Phase 1:Obtaining E-Cash


1.Send E-Cash
2.deliver goods
Merchant
Consumer
Phase 2:Purchasing with E-Cash


1.Redeem e-cash
2.Credit Merchant's account
Electronic Mint
Merchant
Merchant's Bank
Phase 3:Redeeming Cash by the merchant

Appendix B. Questionnaire D

B.1. Self-estimate form

Fill-iner:

Estimate item/method	Conventional requirement representation	Visualized requirement representation
Spent time (min)		
Difficulty in production	☐ very high ☐ high ☐ middle ☐ easy ☐ very easy	☐ very high ☐ high ☐ middle ☐ easy ☐ very easy
If being offered media database for employment, you think the difficulty in production is	☐ very high ☐ high ☐ middle ☐ easy ☐ very easy	☐ very high ☐ high ☐ middle ☐ easy ☐ very easy
If being offered special instrument for requirement representation, you think the help for lowing difficulty in production is	☐ very helpful ☐ helpful ☐ no influence	☐ very helpful ☐ helpful ☐ no influence
Degree to express problems	☐ complete expression ☐ 75% ☐ middle (50%) ☐ 25% ☐ unable to express	☐ complete expression ☐ 75% ☐ middle (50%) ☐ 25% ☐ unable to express
Source of media	☐ self-produced ☐ produced by others ☐ use visible materials ☐ etc.	☐ self-produced ☐ produced by others ☐ use visible materials ☐ etc.
User's communicating ability	☐ very high ☐ high ☐ middle ☐ low ☐ very low	☐ very high ☐ high ☐ middle ☐ low ☐ very low
Understandability	☐ very high ☐ high ☐ middle ☐ low ☐ very low	☐ very high ☐ high ☐ middle ☐ low ☐ very low
Material amount (Kbyte)		
Confront problems and (possible) way to resolve		

If the repetitive use of multimedia elements has to be self-produced and accumulated by you, which method are you willing to adopt?

☐ conventional requirement representation
☐ visualized requirement representation

If the repetitive use of multimedia element base has been established, which method are you willing to adopt?

☐ conventional requirement representation
☐ visualized requirement representation

Appendix C. Questionnaire E

C.1. Crossestimate form

Fill-iner:

Estimate item/method	Conventional requirement representation	Visualized requirement representation
Study time (min)		
Degree to express problems	☐ complete expression (100%) ☐ 75% ☐ middle(50%) ☐ 25% ☐ unable to express	☐ complete expression (100%) ☐ 75% ☐ middle(50%) ☐ 25% ☐ unable to express
User's communicating ability	☐ very high ☐ high ☐ middle ☐ low ☐ very low	☐ very high ☐ high ☐ middle ☐ low ☐ very low
Understandability	☐ very high ☐ high ☐ middle ☐ low ☐ very low	☐ very high ☐ high ☐ middle ☐ low ☐ very low
If the operation of multimedia is advisable	☐ very advisable ☐ advisable ☐ middle ☐ unadvisable ☐ very unadvisable	☐ very advisable ☐ advisable ☐ middle ☐ unadvisable ☐ very unadvisable

If you are a system requester, which method are you willing to adopt?
☐ conventional requirement representation
☐ visualized requirement representation
If you are a system analyst, which method are you willing to adopt?
☐ conventional requirement representation
☐ visualized requirement representation
Do you think using visualized request description is helpful for getting correct software requirement?
☐ yes ☐ no

References

Baroth, C.H., 1995. Visual programming in the real world. In: Burnett, M.M., Goldberg, A., Lewis, T. (Eds.), Visual Object-oriented Programming Concepts and Environments, pp. 21–42.

Booch, G., 1991. Object-oriented Design with Applications. Benjamin/Commings Publishing Company Inc., Menlo Park, CA.

Breen, D.E., Getto, P.H., Apocada, A.A., Schmidt, D.G., Sarachan, B.D., 1987. The clockworks: an object-oriented computer animation system. In: Proceedings of Eurographics, pp. 275–282.

Burnett, M.M., Baker, M.J., Bohus, C., Carlson, P., Yang, S., van Zee, P., 1995. Scaling up visual programming languages. IEEE Computer (March) 45–54.

Chang, S.K., Costagliola, G., Pacini, G., 1995. Visual-language system for user interfaces. IEEE Software 12 (2), 33–44.

Chen, W.C., 1998. A visual and reuse-based paradigm for software construction. Ph.D. Dissertation, Computer Science and Information Engineering Department, National Chiao Tung University, Taiwan.

Chorng-Shiuh, K., 1995. The design and implementation of a script language and playback system for electronic dtory book. Master Thesis of National Chiao Tung University, Taiwan.

Coad, P., Yourdon, E., 1990. Object-oriented Analysis. Prentice-Hall, Englewood Cliffs, NJ.

Freeman, P., 1987. A Perspective on Reusability. The Computer Society of the IEEE, pp. 2–8.

Goguen, J.A., 1996. Formality and informality in requirements engineering. In: Proceedings of the 2nd International Conference on Requirements Engineering, April 15–18, Colorado Springs, Colorado, pp. 102–108.

Hirakawa, M., Iwata, S., Yoshimoto, I., Tanaka, M., Ichidawa, T., 1987. Hi-visual iconic programming. In: Proceedings of the IEEE Workshop Visual Language, pp. 305–314.

Jirotka, M., Heath, C., Luff, P., 1995. Ethnography by video for requirements capture. In: Proceedings of the 2nd International Symposium on Requirements Engineering, April 27–29, York, England, pp. 190–191.

Keepence, B., Mannion, M., Smith, S., 1995. SMARTRe requirements: writing reusable requirements. In: Proceedings of the 2nd International Symposium on Requirements Engineering, April 27–29, York, England, pp. 27–35.

Lam, W., McDermid, J.A., Vickers, A.J., 1997. Ten steps towards systematic requirements reuse. In: Proceedings of the 3rd International Symposium on Requirements Engineering, July 6–10, 1997, Annapolis, MD, USA, pp. 6–15.

Lenz, M., Schmid, H.A., Wolf, P.W., 1987. Software reuse through building blocks. IEEE Software, 34–42.

Li, C.L., 1992. An object-based icon programming methodology. Master Thesis of National Chiao Tung University, Taiwan, June.

Lubbars, M.D., 1987. Wide-spectrum support for software reusability. In: Proceedings of the Workshop on Software Reusability and Maintainability, October.

Maien, N.A.M., Mistry, P., Sutcliffe, A.G., 1995. How people categorise requirements for reuse: a natural approach. In: Proceedings of the 2nd International Symposium on Requirements Engineering, April 27–29, York, England, pp. 148–155.

Massonet, P., van Lamsweerde, A., 1997. Analogical reuse of requirements frameworks. In: Proceedings of the 3rd International Symposium on Requirements Engineering, July 6–10, Annapolis, Maryland, USA, pp. 26–37.

Ohnishi, A., 1994. A visual software requirements definition method. In: IEEE Proceedings: The first International Conference on Requirements Engineering, April 18–22, Colorado Springs, Colorado, pp. 194–201.

Pressman, R.S., 1992. Software Engineering – A Practitioner's Approach, third ed. McGraw-Hill, New York.

Rumbaugh, J. et al., 1987. Object-oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, NJ.

Takahashi, K., Potts, C., Kumar, V., Ota, K., Smith, J.D., 1996. Hypermedia support for collaboration in requirements analysis. In: Proceedings of the 2nd International Conference on Requirements Engineering, April 15–18, Colorado Springs, Colorado, pp. 31–40.

Wood, D.P., Christel, M.G., Stevens, S.M., 1994. A multimedia approach to requirements capteru and modeling. In: IEEE Proceedings: The first International Conference on Requirements Engineering, April 18–22, Colorado Springs, Colorado, pp. 53–56.

Dr. Krishna M. Kavi is currently a Professor and Eminent Scholar of Computer Engineering. Prior to joining UAH, he was a Professor of Computer Science and Engineering at the University of Texas at Arlington. For two years (1993–1995) he was a Program Manager at the National Science Foundation, managing Operating Systems, and Programming Languages and Compilers programs in CCR Division. He was an IEEE Computer Society (CS) Distinguished Visitor (1989–1991), Editor of the IEEE Transactions on Computers (1993–1997), and Editor of the Computer Society Press (1987–1991). His primary research interest lies in Computer Systems Architecture, including dataflow and multithreaded systems, Memory management, Operating Systems, and Compiler Optimization. His other research interests include Formal specification of Concurrent Processing Systems, Performance Modeling and Evaluation, Load Balancing and Scheduling of Parallel Programs. He published over 125 technical papers on these topics. He received his B.E. (Electrical) from the Indian Institute of Science, MS and Ph.D. (Computer Science and Engineering) from the Southern Methodist University. He is a Senior Member of the IEEE and a member of the ACM.

Wu-Chi Chen received his B.S. and Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University (Hsinchu, Taiwan) in 1992 and 1998. Since then he joined the Taiwan Semiconductor Manufacturing Co. as a Section Manager. His research interests include Software Engineering, Object-oriented Modeling, CIM System Integration, Supply Chain Management, and Factory Planning System.

Deng-Jyi Chen received the B.S. degree in Computer Science from Missouri State University (cape Girardeau), USA, and M.S. and Ph.D. degree in Computer Science from the University of Texas (Arlington), USA in 1983, 1985, 1988, respectively.

He is now a professor at Computer Science and Information Engineering Department of National Chiao Tung University (Hsinchu, Taiwan). Prior to joining the faculty of National Chiao Tung University, he was with National Cheng Kung University (Tainan, Taiwan). So far, he has been publishing *more than 100* referred papers in the area of performance and reliability modeling and evaluation of distributed systems, computer networks, fault-tolerant system, software reuse, object-oriented systems, and multimedia application systems. He has been invited to talk and to present papers around the world (USA, Canada, UK, Japan, Korea, China, Hong Kong, Netherlands, Germany, Switzerland, Spain, and Italy). Some of his research results have been technology transferred to some companies and used in product design and implementation. So far, he has been a chief project leader of several commercial products. Some of these products are widely used in primary schools for CAI educational tools in Taiwan.

He has also received the research award yearly from National Science Council Taiwan for the past 12 years and serves as a committee member in several academic and industrial organizations.